Exhibit 5

MicroPairing Technologies LLC
US Patent No. 7,178,049

Method for Multi-Tasking Multiple Java Virtual Machines in a Secure Environment

Claim 29

Nissan Rogue

AUTOSAR

Claim 29

29. A method for configuring real-time vehicle applications in a distributed multi-processor system operating in a vehicle, comprising:

identifying vehicle applications running on different processors in the multiprocessor system;

operating a task manager that obtains different data and state information associated with the different vehicle applications;

operating a configuration manager that notifies the task manager upon detecting a failure running one of the identified vehicle applications in the multiprocessor system;

using the task manager for automatically identifying another processor in the multiprocessor system for running the identified vehicle application and redirecting the vehicle application associated with the detected failure to the other identified processor in the vehicle;

using the configuration manager to redirect the data and state information to the other identified processor in the vehicle after detecting the failure; and

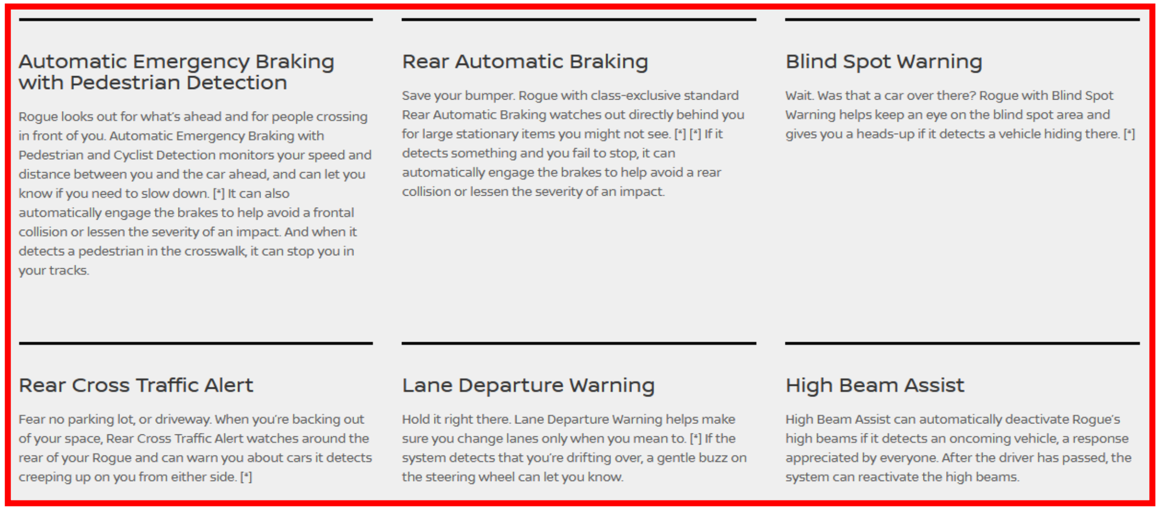
initiating the identified application in the identified other processor.

Source: Claim 31 of US Patent 7,178,049

29. A method for configuring real-time vehicle applications in a distributed multi-processor system operating in a vehicle, comprising:

Note: Nissan Rogue comprises a host of advanced features, many of which are related to safety and require assurance of their proper operation. Nissan Rogue comprises many features necessarily requiring a real-time distributed multiprocessor system. AUTOSAR comprises the tools necessary to configure individual ECUs as well as a system of ECU applications. Nissan is a Premium Partner in the AUTOSAR consortium.

Nissan Rogue - AUTOSAR



Automatic Emergency Braking with Pedestrian Detection

Rogue looks out for what's ahead and for people crossing in front of you. Automatic Emergency Braking with Pedestrian and Cyclist Detection monitors your speed and distance between you and the car ahead, and can let you know if you need to slow down. [*] It can also automatically engage the brakes to help avoid a frontal collision or lessen the severity of an impact. And when it detects a pedestrian in the crosswalk, it can stop you in your tracks.

Rear Automatic Braking

Save your bumper. Rogue with class-exclusive standard Rear Automatic Braking watches out directly behind you for large stationary items you might not see. [*] [*] If it detects something and you fail to stop, it can automatically engage the brakes to help avoid a rear collision or lessen the severity of an impact.

Blind Spot Warning

Wait. Was that a car over there? Rogue with Blind Spot Warning helps keep an eye on the blind spot area and gives you a heads-up if it detects a vehicle hiding there. [*]

Rear Cross Traffic Alert

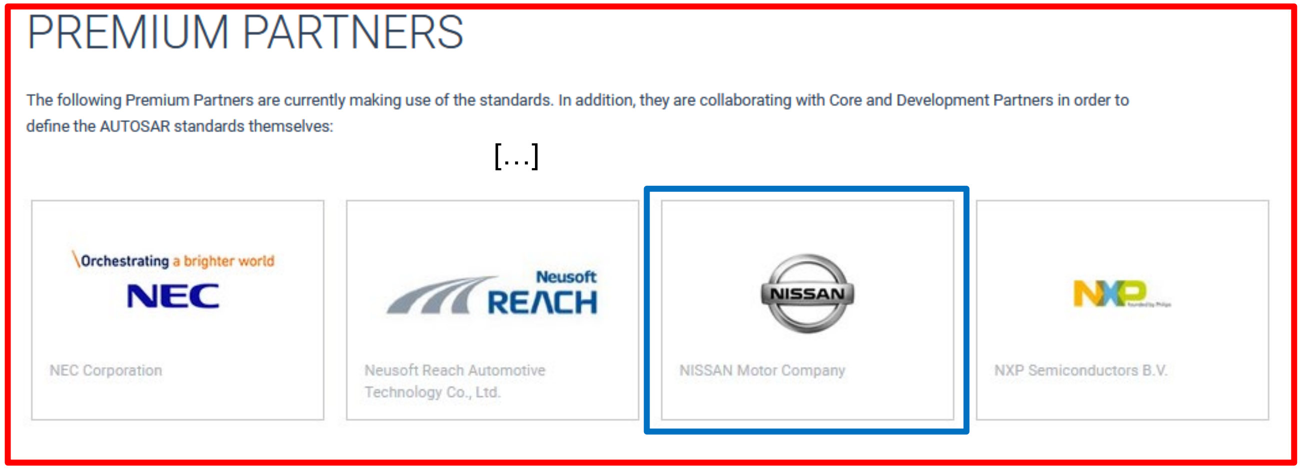
Fear no parking lot, or driveway. When you're backing out of your space, Rear Cross Traffic Alert watches around the rear of your Rogue and can warn you about cars it detects creeping up on you from either side. [*]

Lane Departure Warning

Hold it right there. Lane Departure Warning helps make sure you change lanes only when you mean to. [*] If the system detects that you're drifting over, a gentle buzz on the steering wheel can let you know.

High Beam Assist

High Beam Assist can automatically deactivate Rogue's high beams if it detects an oncoming vehicle, a response appreciated by everyone. After the driver has passed, the system can reactivate the high beams.



PREMIUM PARTNERS

The following Premium Partners are currently making use of the standards. In addition, they are collaborating with Core and Development Partners in order to define the AUTOSAR standards themselves:

[…]

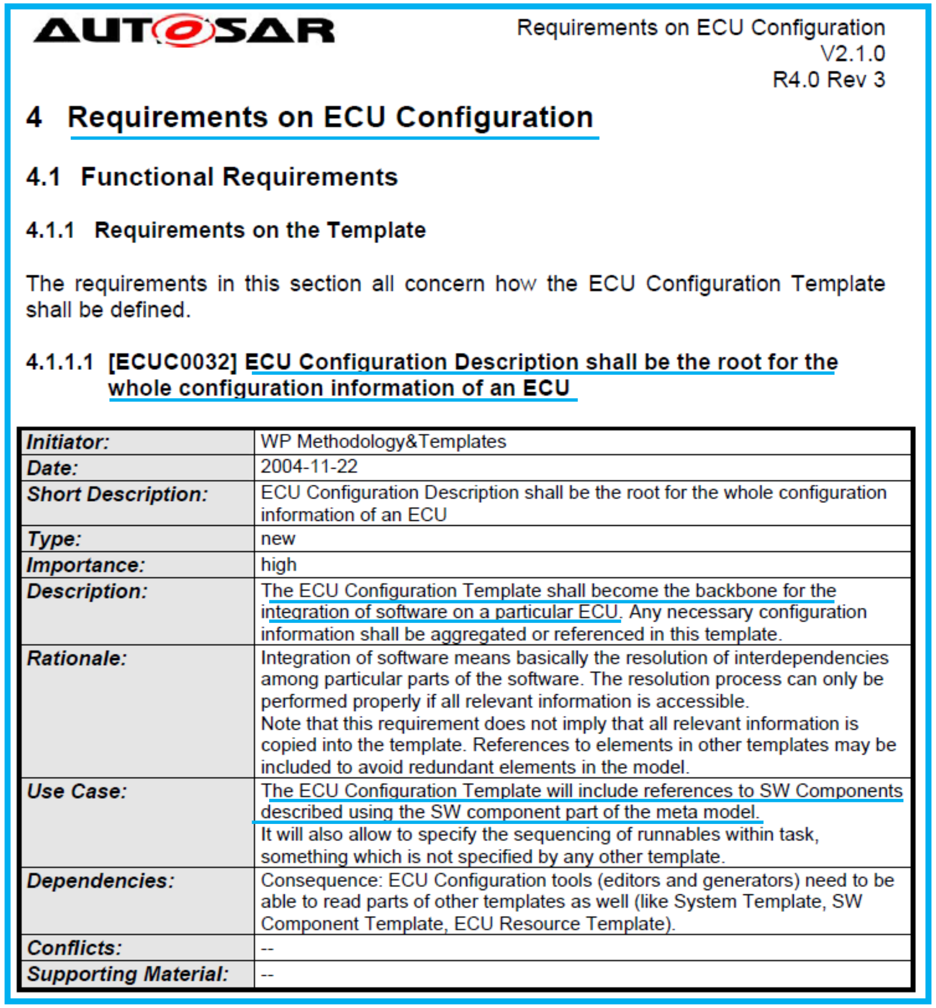
Orchestrating a brighter world NEC	Neusoft REACH	NISSAN	NXP
NEC Corporation	Neusoft Reach Automotive Technology Co., Ltd.	NISSAN Motor Company	NXP Semiconductors B.V.

Source: (top) https://www.nissanusa.com/vehicles/crossovers-suvs/rogue/features/safety-driver-assist.html
(bottom) https://www.autosar.org/about/current-partners/premium-partners/

29. A method for configuring real-time vehicle applications in a distributed multi-processor system operating in a vehicle, comprising:

Note: AUTOSAR requires ECU configuration. Templates are defined that include references to software components and their operations. Tools including editors and generators are available for editing configurations for individual ECUs as well as a system of ECU applications.

Nissan Rogue - AUTOSAR



AUTOSAR

Requirements on ECU Configuration
V2.1.0
R4.0 Rev 3

4 Requirements on ECU Configuration

4.1 Functional Requirements

4.1.1 Requirements on the Template

The requirements in this section all concern how the ECU Configuration Template shall be defined.

4.1.1.1 [ECUC0032] ECU Configuration Description shall be the root for the whole configuration information of an ECU

Initiator:	WP Methodology&Templates
Date:	2004-11-22
Short Description:	ECU Configuration Description shall be the root for the whole configuration information of an ECU
Type:	new
Importance:	high
Description:	The ECU Configuration Template shall become the backbone for the integration of software on a particular ECU. Any necessary configuration information shall be aggregated or referenced in this template.
Rationale:	Integration of software means basically the resolution of interdependencies among particular parts of the software. The resolution process can only be performed properly if all relevant information is accessible. Note that this requirement does not imply that all relevant information is copied into the template. References to elements in other templates may be included to avoid redundant elements in the model.
Use Case:	The ECU Configuration Template will include references to SW Components described using the SW component part of the meta model. It will also allow to specify the sequencing of runnables within task, something which is not specified by any other template.
Dependencies:	Consequence: ECU Configuration tools (editors and generators) need to be able to read parts of other templates as well (like System Template, SW Component Template, ECU Resource Template).
Conflicts:	--
Supporting Material:	--

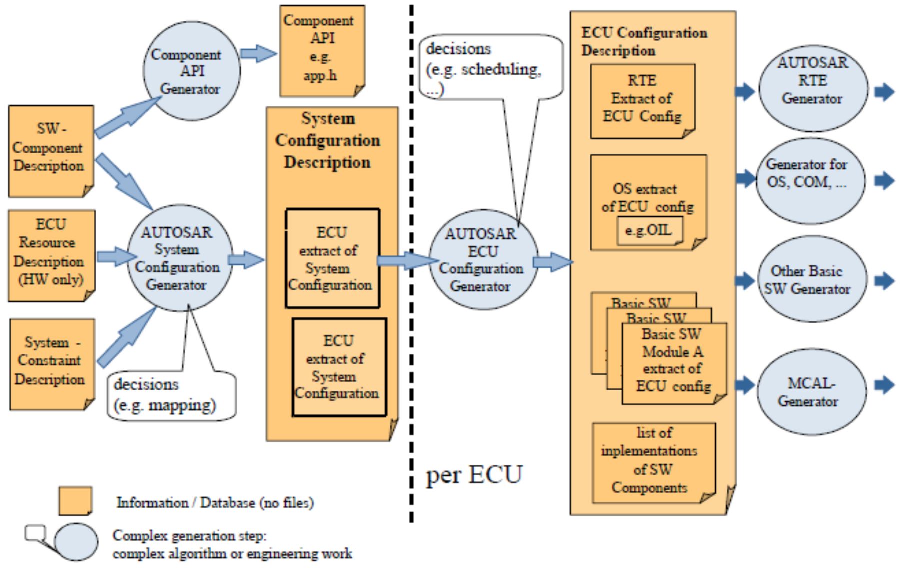
Source: Requirements on ECU Configuration, AUTOSAR Document No. 085, R 4.0 Rev 3.

29. A method for configuring real-time vehicle applications in a distributed multi-processor system operating in a vehicle, comprising:

Note: AUTOSAR provides a methodology for configuring ECUs, software components and system constraints.

Nissan Rogue - AUTOSAR

System Template
V4.2.0
R4.0 Rev 3



SW-Component Description
ECU Resource Description (HW only)
System-Constraint Description
Component API Generator
Component API e.g. app.h
AUTOSAR System Configuration Generator
decisions (e.g. mapping)
System Configuration Description
ECU extract of System Configuration
ECU extract of System Configuration
decisions (e.g. scheduling, ...)
AUTOSAR ECU Configuration Generator
per ECU
ECU Configuration Description
RTE Extract of ECU Config
OS extract of ECU config
e.g OIL
Basic SW
Basic SW
Basic SW Module A extract of ECU config
list of implementations of SW Components
AUTOSAR RTE Generator
Generator for OS, COM, ...
Other Basic SW Generator
MCAL-Generator
Information / Database (no files)
Complex generation step: complex algorithm or engineering work

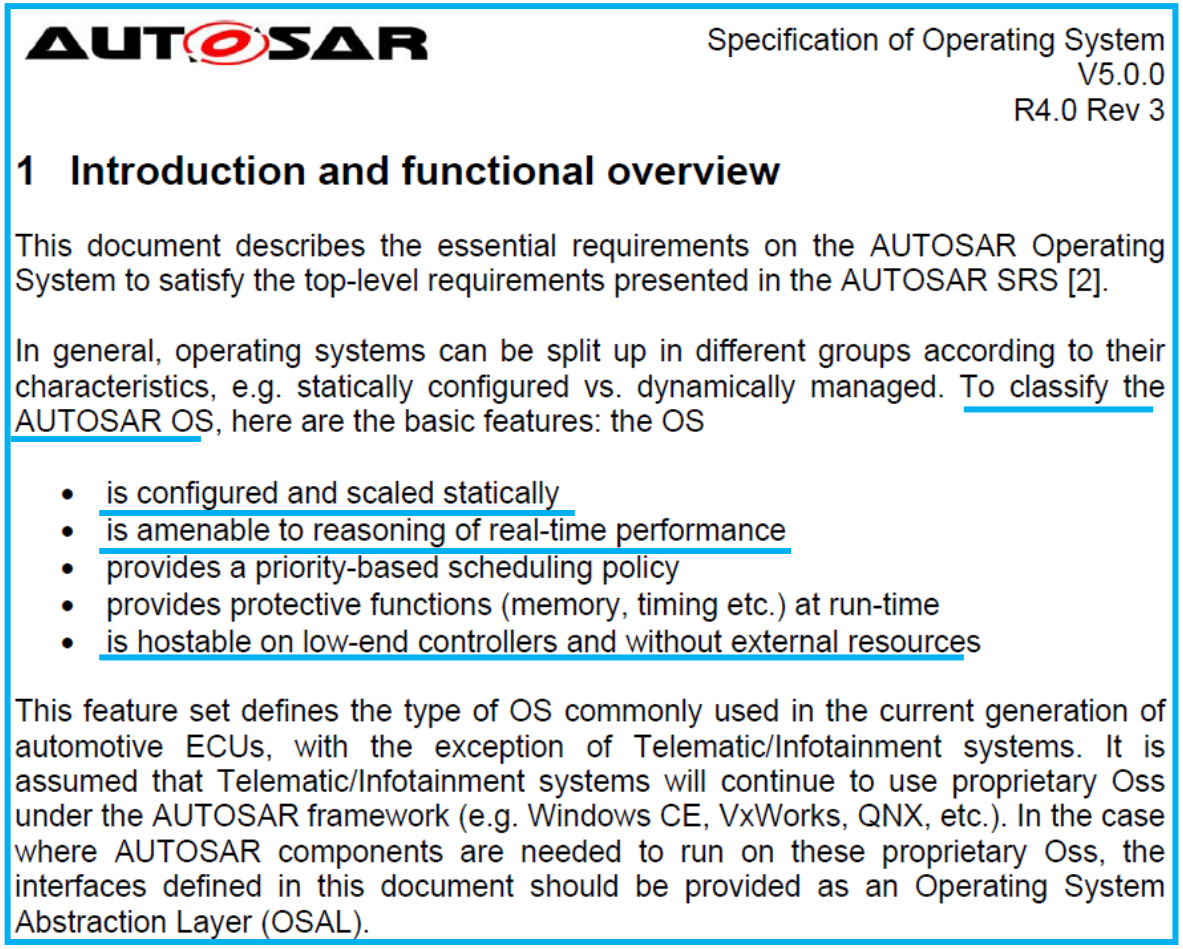
Figure 1.2: AUTOSAR Methodology

Source: 1. System Template, AUTOSAR Document ID 063, R 4.0 Rev 3

29. A method for configuring real-time vehicle applications in a distributed multi-processor system operating in a vehicle, comprising:

Note: AUTOSAR requires a real-time operating system for its ECUs.

Nissan Rogue - AUTOSAR


AUTOSAR

Specification of Operating System
V5.0.0
R4.0 Rev 3

1 Introduction and functional overview

This document describes the essential requirements on the AUTOSAR Operating System to satisfy the top-level requirements presented in the AUTOSAR SRS [2].

In general, operating systems can be split up in different groups according to their characteristics, e.g. statically configured vs. dynamically managed. To classify the AUTOSAR OS, here are the basic features: the OS

- is configured and scaled statically
- is amenable to reasoning of real-time performance
- provides a priority-based scheduling policy
- provides protective functions (memory, timing etc.) at run-time
- is hostable on low-end controllers and without external resources

This feature set defines the type of OS commonly used in the current generation of automotive ECUs, with the exception of Telematic/Infotainment systems. It is assumed that Telematic/Infotainment systems will continue to use proprietary Oss under the AUTOSAR framework (e.g. Windows CE, VxWorks, QNX, etc.). In the case where AUTOSAR components are needed to run on these proprietary Oss, the interfaces defined in this document should be provided as an Operating System Abstraction Layer (OSAL).

Source: Specification of Operating System, AUTOSAR Document ID 034, R 4.0 Rev 3

29. A method for configuring real-time vehicle applications in a distributed multi-processor system operating in a vehicle, comprising:

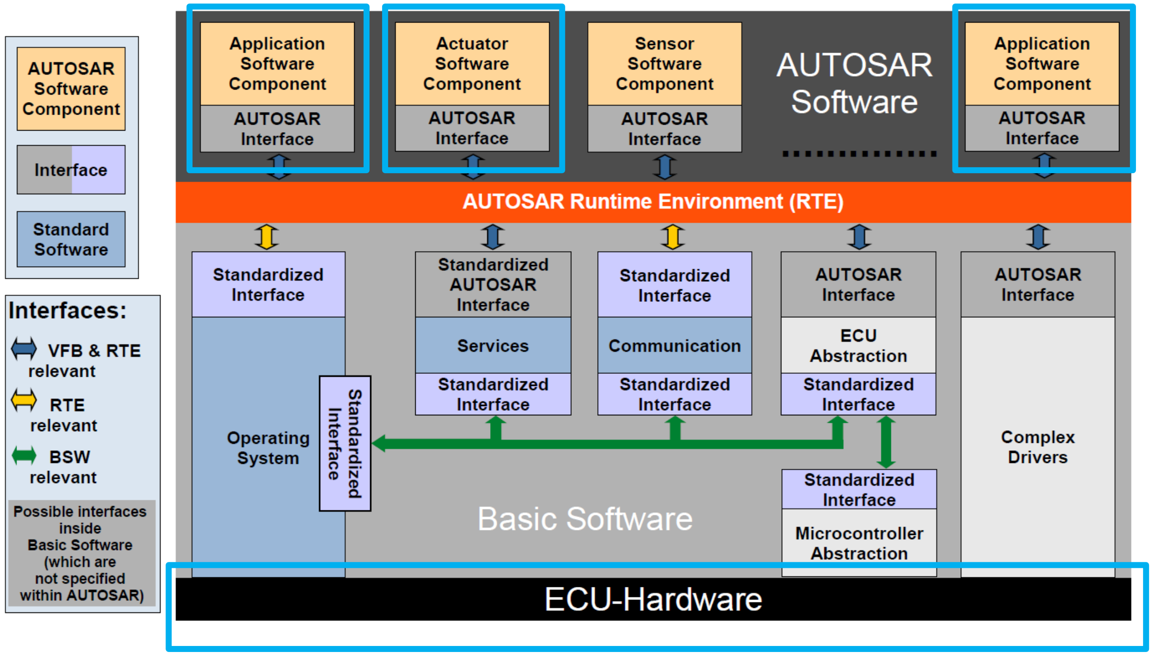
Note: AUTOSAR requires a real-time operating system for its ECUs. AUTOSAR provides the capability to distribute real-time applications on one ECU.

Nissan Rogue - AUTOSAR

Application scope of AUTOSAR

AUTOSAR is dedicated for Automotive ECUs. Such ECUs have the following properties:

- strong interaction with hardware (sensors and actuators),
- connection to vehicle networks like CAN, LIN, FlexRay or Ethernet,
- microcontrollers (typically 16 or 32 bit) with limited resources of computing power and memory (compared with enterprise solutions),
- Real Time System and
- program execution from internal or external flash memory.



Source: Layered Software Architecture, AUTOSAR Document ID 053, R 4.0 Rev 3, pages 17 (top) and 68 (bottom).

Nissan Rogue - AUTOSAR

29. A method for configuring real-time vehicle applications in a distributed multi-processor system operating in a vehicle, comprising:

Note: AUTOSAR provides for the integration of multiple ECUs (electronic control units) into one system. Each ECU has its own processor, software components (SW-C), Run -Time Environment (RTE), and Basic Software (BSW). The distributed ECUs (micro-processors) communicate over communication busses.

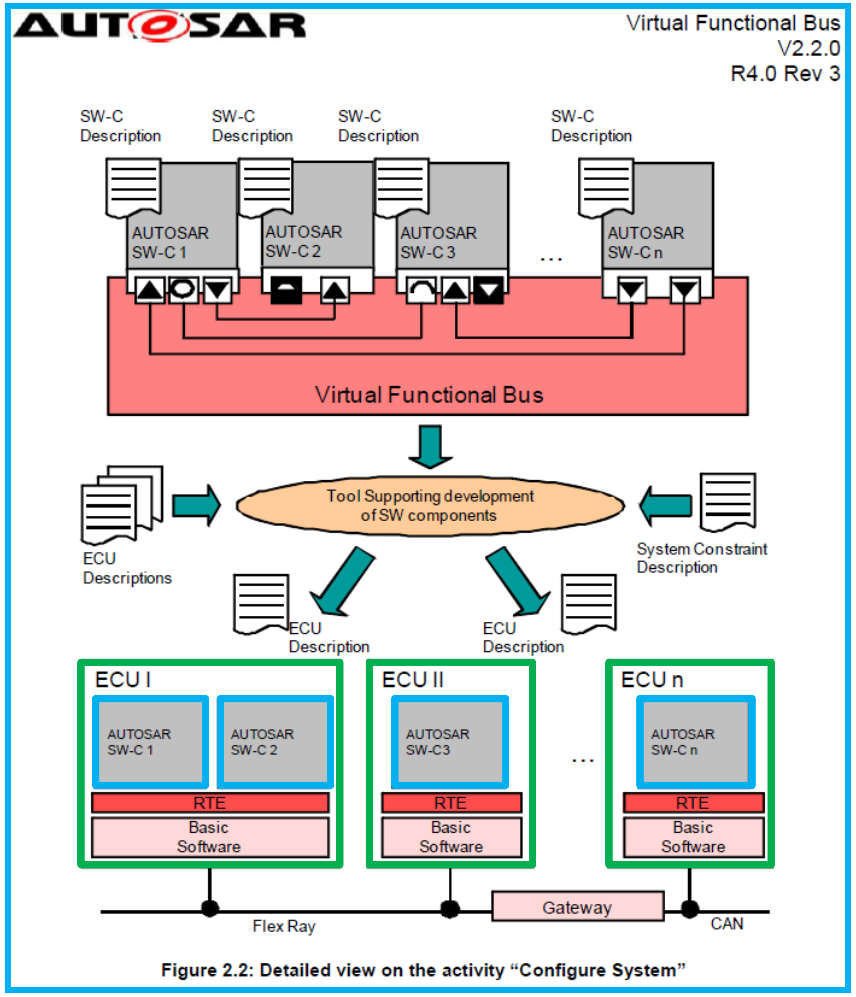


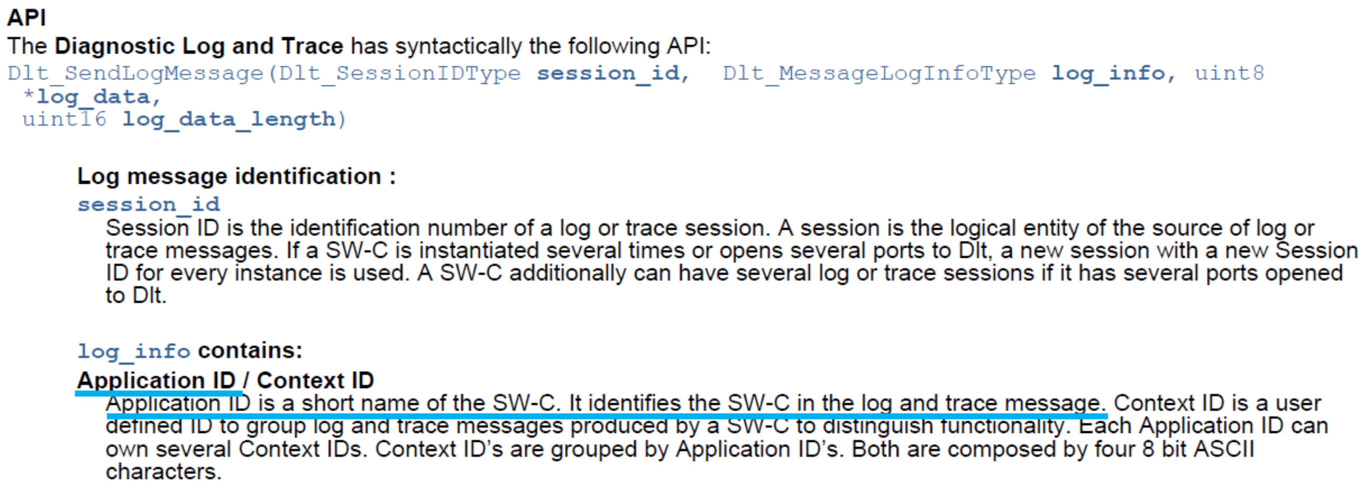
Figure 2.2: Detailed view on the activity "Configure System"

Source: Virtual Functional Bus, AUTOSAR Document ID 056, R 4.0 Rev 3.

identifying vehicle applications running on different processors in the multiprocessor system;

Note: AUTOSAR provides templates and tools to configure ECUs and software application (see slides 4 and 5). Software applications are identified in various ways. For example, the application has an ID for Diagnostic Log and Trace. The application also has an ID to be used by the operating system for sending inter operating system communication (IoC) messages (next slide).

Nissan Rogue - AUTOSAR



API

The **Diagnostic Log and Trace** has syntactically the following API:

```
Dlt_SendLogMessage(Dlt_SessionIDType session_id,  Dlt_MessageLogInfoType log_info, uint8
 *log_data,
 uint16 log_data_length)
```

Log message identification :

`session_id`

Session ID is the identification number of a log or trace session. A session is the logical entity of the source of log or trace messages. If a SW-C is instantiated several times or opens several ports to Dlt, a new session with a new Session ID for every instance is used. A SW-C additionally can have several log or trace sessions if it has several ports opened to Dlt.

`log_info` **contains:**

Application ID / Context ID

Application ID is a short name of the SW-C. It identifies the SW-C in the log and trace message. Context ID is a user defined ID to group log and trace messages produced by a SW-C to distinguish functionality. Each Application ID can own several Context IDs. Context ID's are grouped by Application ID's. Both are composed by four 8 bit ASCII characters.

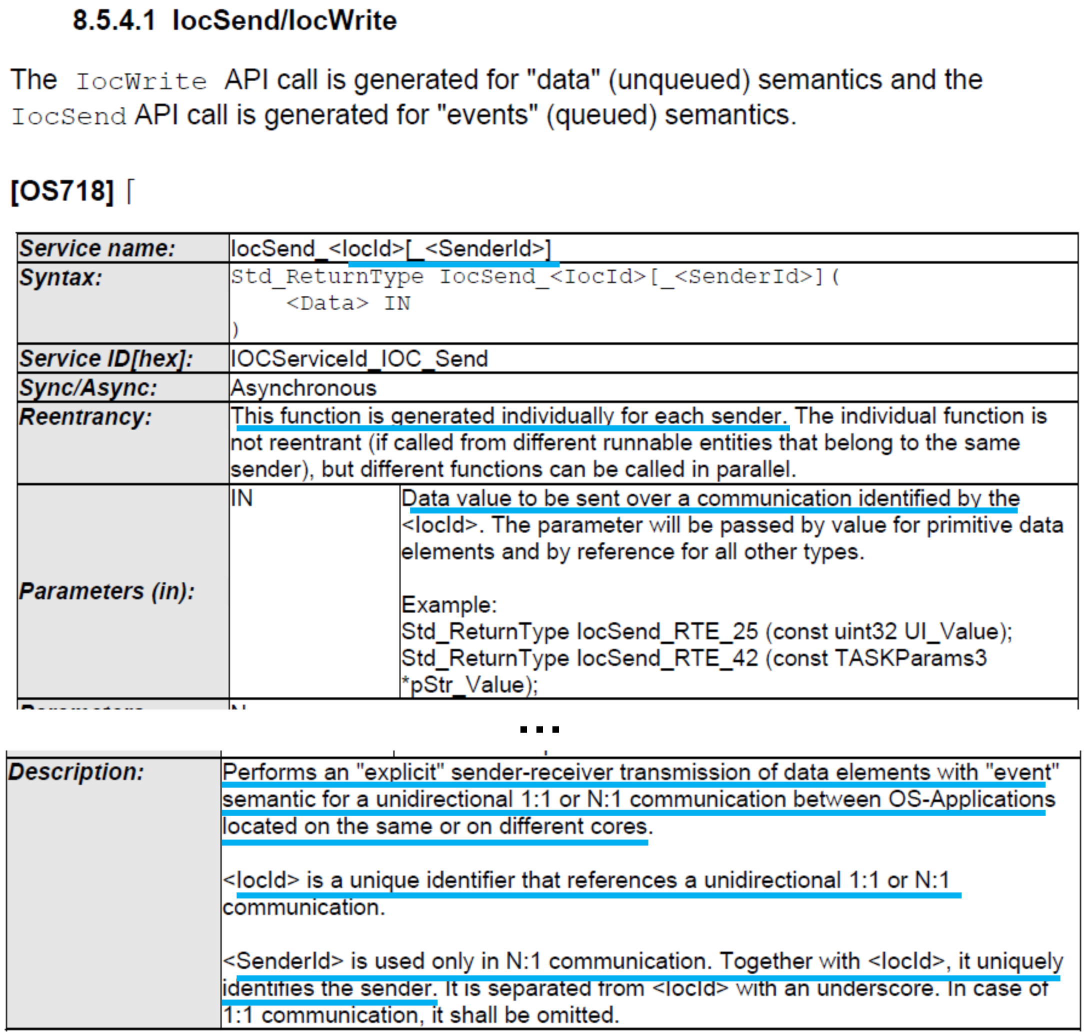
Source: Layered Software Architecture, AUTOSAR Document ID 053, R 4.0 Rev 3, page 135.

nelson bumgardner albritton

identifying vehicle applications running on different processors in the multiprocessor system;

Note: AUTOSAR provides templates and tools to configure ECUs and software application (see slides 4 and 5). Software applications are identified in various ways. For example, the application has an ID for Diagnostic Log and Trace (see slide 9). The application also has an ID to be used by the operating system for sending inter operating system communication (IoC) messages.

Nissan Rogue - AUTOSAR



8.5.4.1 IocSend/IocWrite

The `IocWrite` API call is generated for "data" (unqueued) semantics and the `IocSend` API call is generated for "events" (queued) semantics.

[OS718] ⌈

Service name:	IocSend_<IocId>[_<SenderId>]	
Syntax:	`Std_ReturnType IocSend_<IocId>[_<SenderId>]( <Data> IN )`	
Service ID[hex]:	IOCServiceId_IOC_Send	
Sync/Async:	Asynchronous	
Reentrancy:	This function is generated individually for each sender. The individual function is not reentrant (if called from different runnable entities that belong to the same sender), but different functions can be called in parallel.	

Parameters (in):	IN	Data value to be sent over a communication identified by the <IocId>. The parameter will be passed by value for primitive data elements and by reference for all other types.
		
		Example:
		Std_ReturnType IocSend_RTE_25 (const uint32 UI_Value);
		Std_ReturnType IocSend_RTE_42 (const TASKParams3 *pStr_Value);

. . .

Description:	Performs an "explicit" sender-receiver transmission of data elements with "event" semantic for a unidirectional 1:1 or N:1 communication between OS-Applications located on the same or on different cores. <IocId> is a unique identifier that references a unidirectional 1:1 or N:1 communication. <SenderId> is used only in N:1 communication. Together with <IocId>, it uniquely identifies the sender. It is separated from <IocId> with an underscore. In case of 1:1 communication, it shall be omitted.

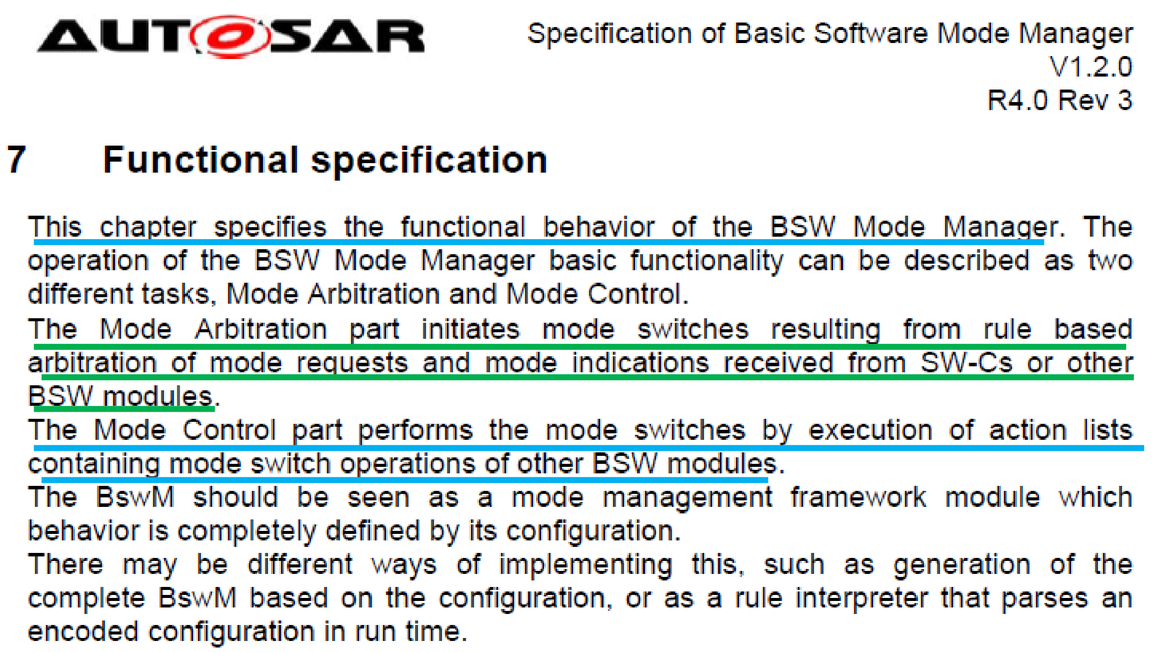
Source: Specification of Operating System, AUTOSAR Document ID 034, R 4.0 Rev 3

nelson bumgardner albritton

operating a task manager that obtains different data and state information associated with the different vehicle applications;

Note: AUTOSAR provides a BSW Mode Manager that functions as a task manager. The BSW Mode manager receives mode requests and indications from applications. The BSW Mode manager executes action lists (tasks) based on mode requests and indications received.

Nissan Rogue - AUTOSAR



AUTOSAR

Specification of Basic Software Mode Manager
V1.2.0
R4.0 Rev 3

7 Functional specification

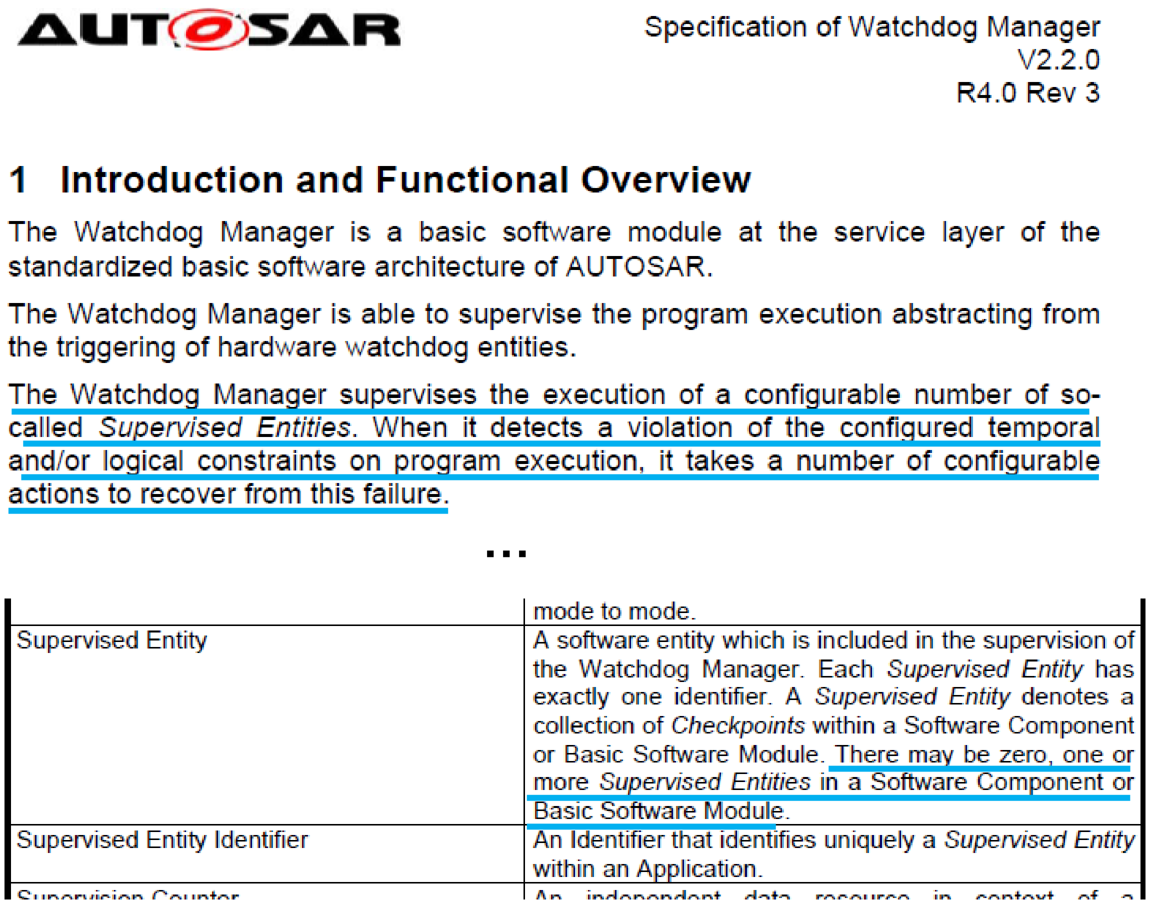
This chapter specifies the functional behavior of the BSW Mode Manager. The operation of the BSW Mode Manager basic functionality can be described as two different tasks, Mode Arbitration and Mode Control.
The Mode Arbitration part initiates mode switches resulting from rule based arbitration of mode requests and mode indications received from SW-Cs or other BSW modules.
The Mode Control part performs the mode switches by execution of action lists containing mode switch operations of other BSW modules.
The BswM should be seen as a mode management framework module which behavior is completely defined by its configuration.
There may be different ways of implementing this, such as generation of the complete BswM based on the configuration, or as a rule interpreter that parses an encoded configuration in run time.

Source: Specification of Basic Software Mode Manager, AUTOSAR Document No. 313, R 4.0 Rev 3

operating a configuration manager that notifies the task manager upon detecting a failure running one of the identified vehicle applications in the multiprocessor system;

Note: The configuration manager is the Watchdog Manager which monitors supervised entities that detect failures.

Nissan Rogue - AUTOSAR



AUTOSAR

Specification of Watchdog Manager
V2.2.0
R4.0 Rev 3

1 Introduction and Functional Overview

The Watchdog Manager is a basic software module at the service layer of the standardized basic software architecture of AUTOSAR.

The Watchdog Manager is able to supervise the program execution abstracting from the triggering of hardware watchdog entities.

The Watchdog Manager supervises the execution of a configurable number of so-called *Supervised Entities*. When it detects a violation of the configured temporal and/or logical constraints on program execution, it takes a number of configurable actions to recover from this failure.

…

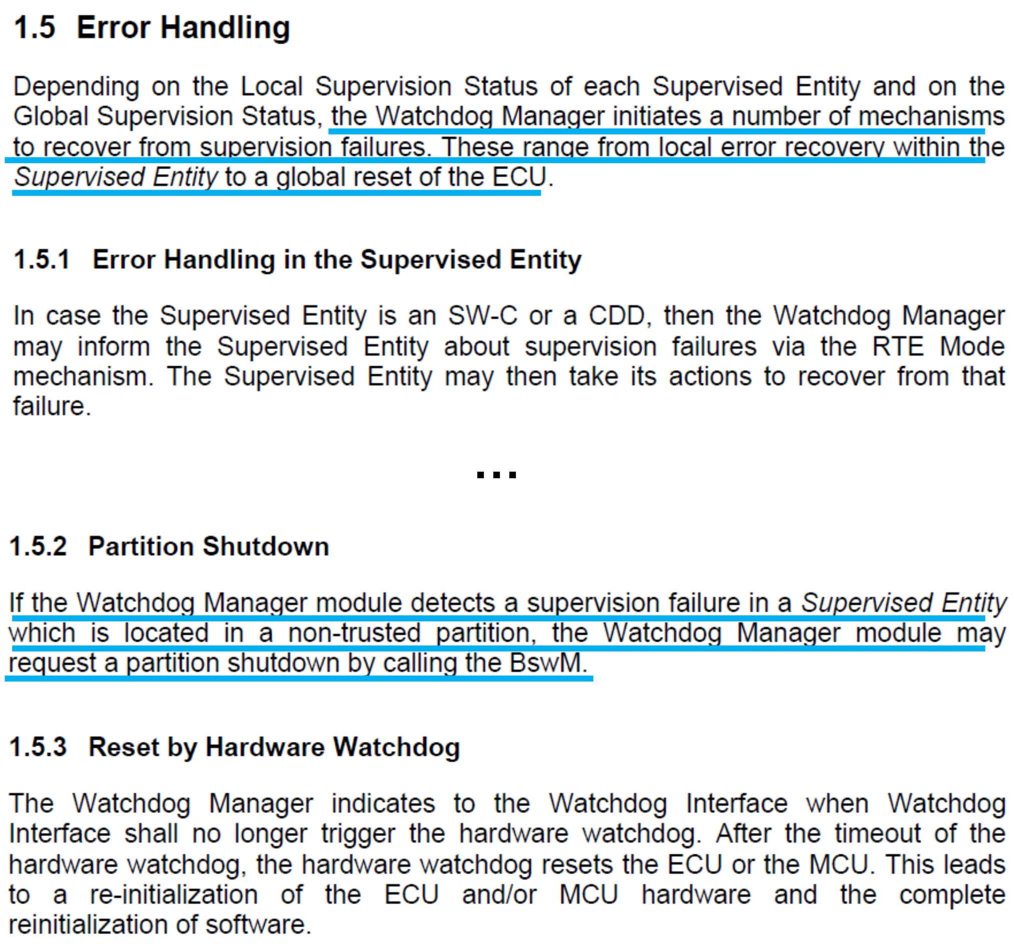
	mode to mode.
Supervised Entity	A software entity which is included in the supervision of the Watchdog Manager. Each *Supervised Entity* has exactly one identifier. A *Supervised Entity* denotes a collection of *Checkpoints* within a Software Component or Basic Software Module. There may be zero, one or more *Supervised Entities* in a Software Component or Basic Software Module.
Supervised Entity Identifier	An Identifier that identifies uniquely a *Supervised Entity* within an Application.
Supervision Counter	An independent data resource in context of a

Source: Specification of Watchdog Manager, AUTOSAR Document ID 080, R 4.0 Rev 3

operating a configuration manager that notifies the task manager upon detecting a failure running one of the identified vehicle applications in the multiprocessor system;

Note: The configuration manager is the Watchdog Manager (see slide 12) that monitors supervised entities (applications), detects failures and enlists one or more of several local actions, including calling the BswM (BSW Mode manager, task manager), to recover from the failure.

Nissan Rogue - AUTOSAR



1.5 Error Handling

Depending on the Local Supervision Status of each Supervised Entity and on the Global Supervision Status, the Watchdog Manager initiates a number of mechanisms to recover from supervision failures. These range from local error recovery within the *Supervised Entity* to a global reset of the ECU.

1.5.1 Error Handling in the Supervised Entity

In case the Supervised Entity is an SW-C or a CDD, then the Watchdog Manager may inform the Supervised Entity about supervision failures via the RTE Mode mechanism. The Supervised Entity may then take its actions to recover from that failure.

...

1.5.2 Partition Shutdown

If the Watchdog Manager module detects a supervision failure in a *Supervised Entity* which is located in a non-trusted partition, the Watchdog Manager module may request a partition shutdown by calling the BswM.

1.5.3 Reset by Hardware Watchdog

The Watchdog Manager indicates to the Watchdog Interface when Watchdog Interface shall no longer trigger the hardware watchdog. After the timeout of the hardware watchdog, the hardware watchdog resets the ECU or the MCU. This leads to a re-initialization of the ECU and/or MCU hardware and the complete reinitialization of software.

Source: Specification of Watchdog Manager, AUTOSAR Document ID 080, R 4.0 Rev 3

operating a configuration manager that notifies the task manager upon detecting a failure running one of the identified vehicle applications in the multiprocessor system;

Note: The configuration manager is the Watchdog Manager has defined interfaces for notifying the task manager (BswM – BSW Mode Manager).

Nissan Rogue - AUTOSAR

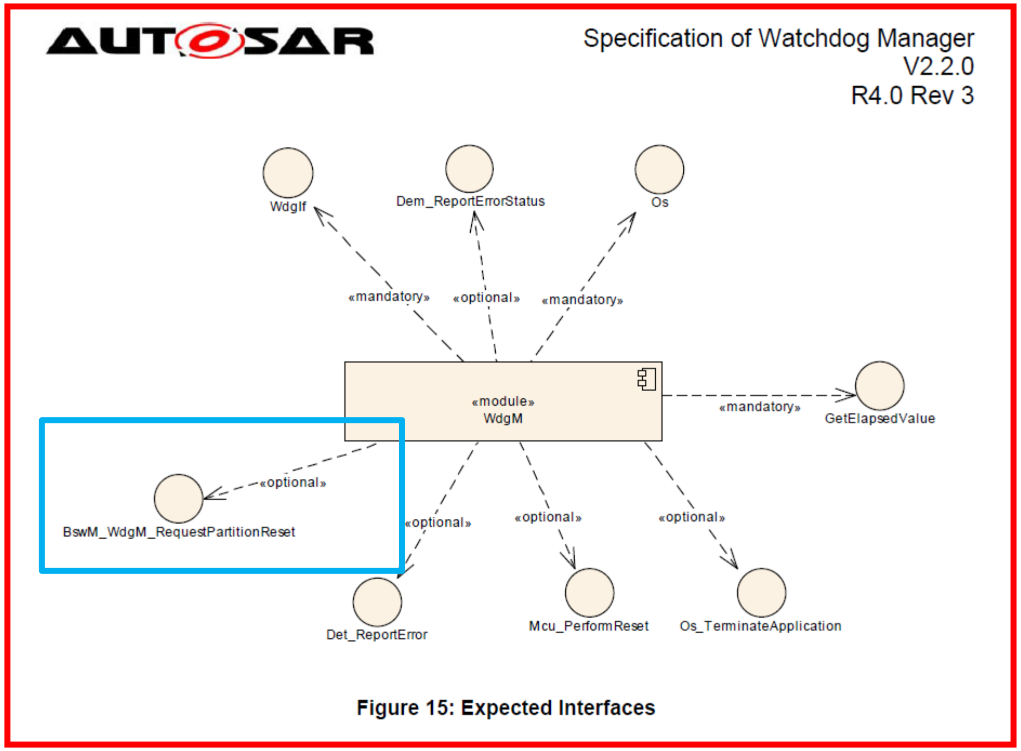


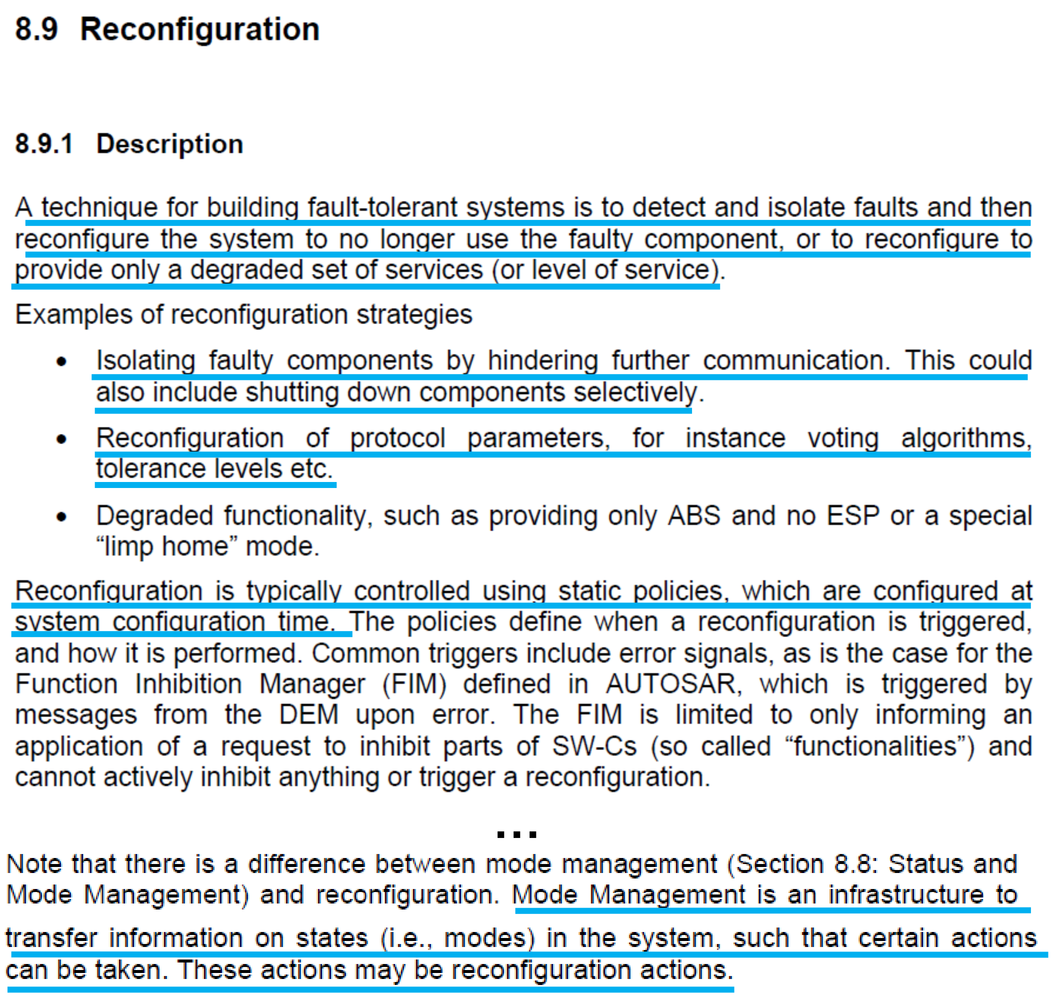
Figure 15: Expected Interfaces

Source: Specification of Watchdog Manager, AUTOSAR Document ID 080, R 4.0

using the task manager for automatically identifying another processor in the multiprocessor system for running the identified vehicle application and redirecting the vehicle application associated with the detected failure to the other identified processor in the vehicle;

Note: AUTOSAR describes and accommodates multiple approaches to fault (due to error) management. For instance, fault-tolerant systems isolate fault components or reconfigure the system as defined during configuration. BSW Mode management (i.e., task manager) is informed on change of states and modes and takes actions that lead to the reconfiguration.

Nissan Rogue - AUTOSAR



8.9 Reconfiguration

8.9.1 Description

A technique for building fault-tolerant systems is to detect and isolate faults and then reconfigure the system to no longer use the faulty component, or to reconfigure to provide only a degraded set of services (or level of service).

Examples of reconfiguration strategies

- Isolating faulty components by hindering further communication. This could also include shutting down components selectively.
- Reconfiguration of protocol parameters, for instance voting algorithms, tolerance levels etc.
- Degraded functionality, such as providing only ABS and no ESP or a special "limp home" mode.

Reconfiguration is typically controlled using static policies, which are configured at system configuration time. The policies define when a reconfiguration is triggered, and how it is performed. Common triggers include error signals, as is the case for the Function Inhibition Manager (FIM) defined in AUTOSAR, which is triggered by messages from the DEM upon error. The FIM is limited to only informing an application of a request to inhibit parts of SW-Cs (so called "functionalities") and cannot actively inhibit anything or trigger a reconfiguration.

…

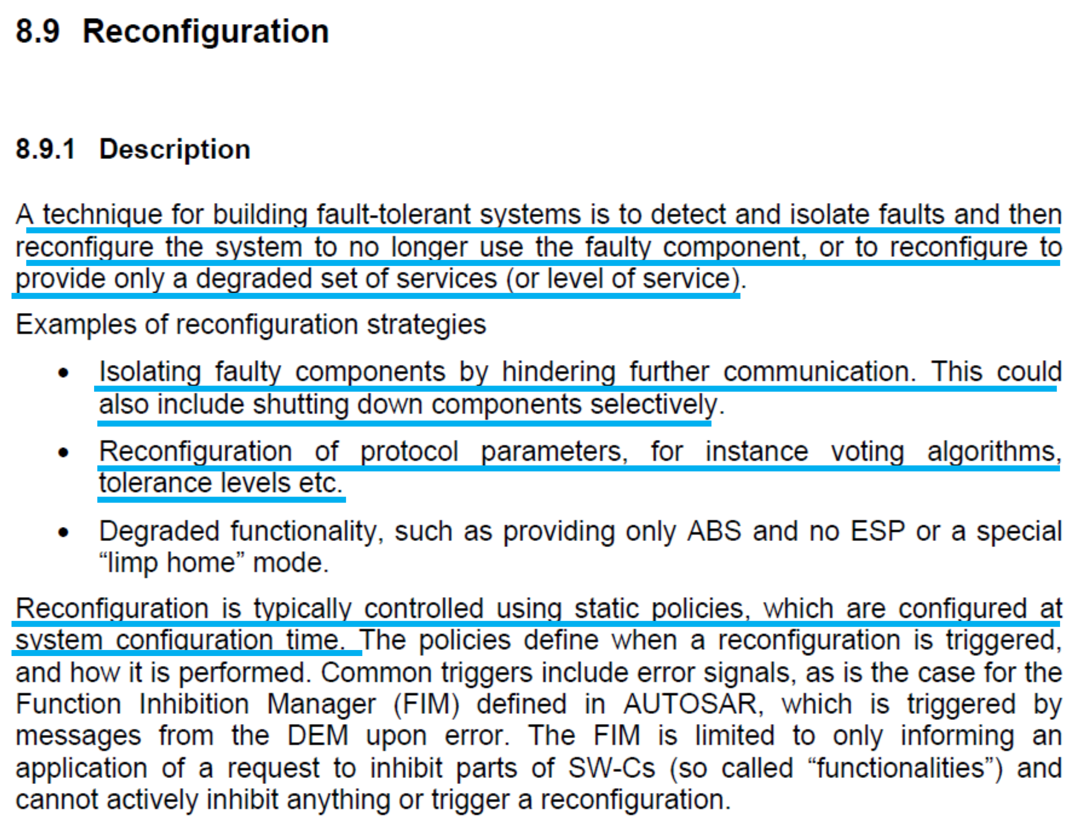
Note that there is a difference between mode management (Section 8.8: Status and Mode Management) and reconfiguration. Mode Management is an infrastructure to transfer information on states (i.e., modes) in the system, such that certain actions can be taken. These actions may be reconfiguration actions.

Source: Explanation of Error Handling on Application Level, AUTOSAR Document No. 378, R 4.0 Rev 1

Nissan Rogue - AUTOSAR

using the configuration manager to redirect the data and state information to the other identified processor in the vehicle after detecting the failure; and

Note: The configuration manger is the Watchdog Manger (see slides 12) that monitors supervised entities (e.g., software applications) and takes action according to the type of error (see slide 13). For instance, fault-tolerant systems isolate fault components or reconfigure the system as defined during configuration. BSW Mode management (i.e., task manager) is informed on change of states and modes and takes actions that lead to the reconfiguration.



8.9 Reconfiguration

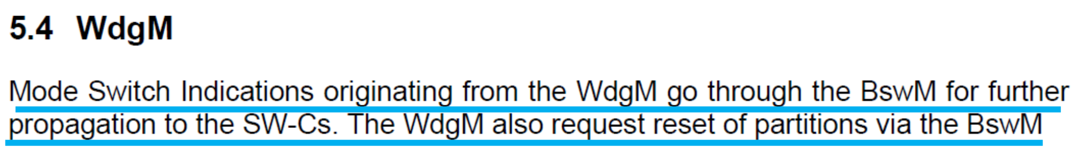
8.9.1 Description

A technique for building fault-tolerant systems is to detect and isolate faults and then reconfigure the system to no longer use the faulty component, or to reconfigure to provide only a degraded set of services (or level of service).

Examples of reconfiguration strategies

- Isolating faulty components by hindering further communication. This could also include shutting down components selectively.
- Reconfiguration of protocol parameters, for instance voting algorithms, tolerance levels etc.
- Degraded functionality, such as providing only ABS and no ESP or a special "limp home" mode.

Reconfiguration is typically controlled using static policies, which are configured at system configuration time. The policies define when a reconfiguration is triggered, and how it is performed. Common triggers include error signals, as is the case for the Function Inhibition Manager (FIM) defined in AUTOSAR, which is triggered by messages from the DEM upon error. The FIM is limited to only informing an application of a request to inhibit parts of SW-Cs (so called "functionalities") and cannot actively inhibit anything or trigger a reconfiguration.



5.4 WdgM

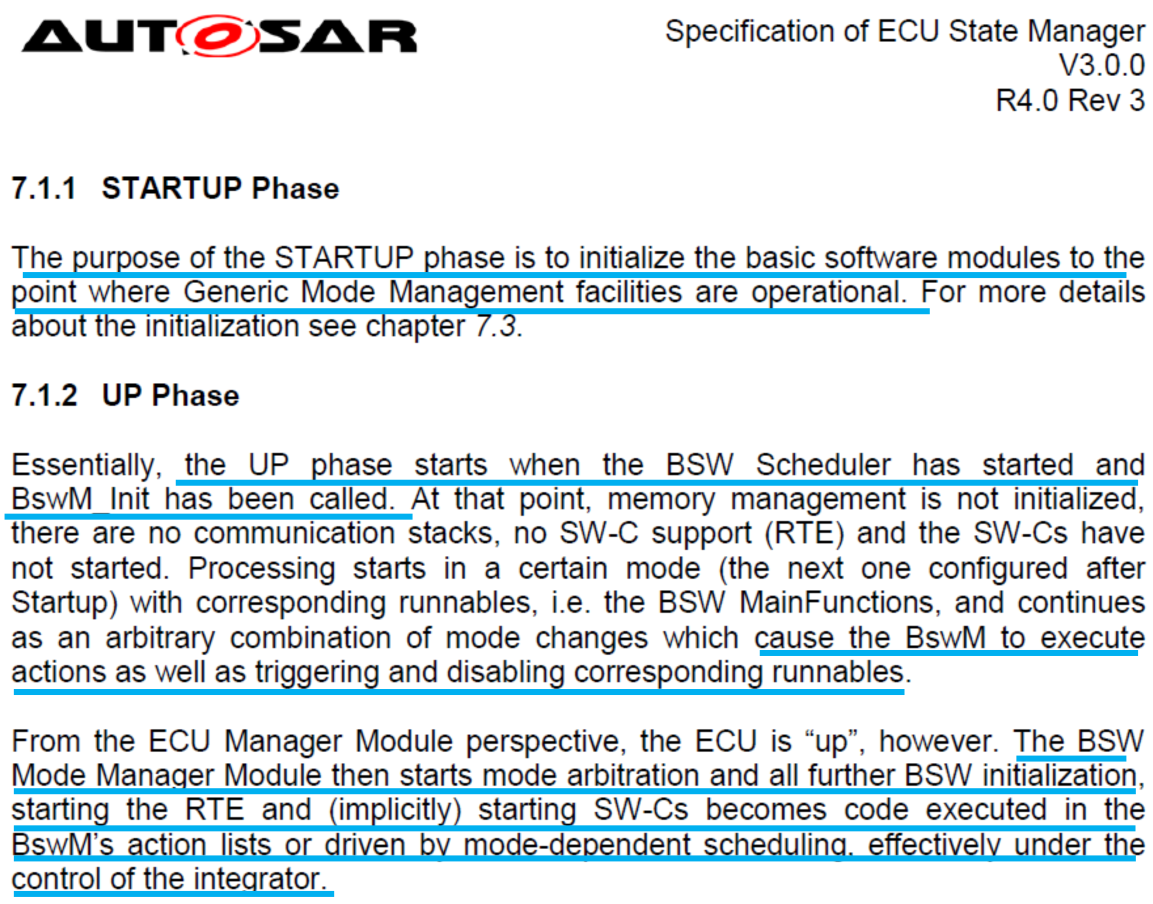
Mode Switch Indications originating from the WdgM go through the BswM for further propagation to the SW-Cs. The WdgM also request reset of partitions via the BswM

Source: (top) Explanation of Error Handling on Application Level, AUTOSAR Document No. 378, R 4.0 Rev 1
(bottom) Specification of Basic Software Mode Manager, AUTOSAR Document No. 313, R 4.0 Rev 3

initiating the identified application in the identified other processor.

Note: *Finally, the ECU State manager starts initialization in the ECU's STARTUP phase. The ECU is considered in the UP phase when the BSW Scheduler has started and the BswM_Init (task manager) has been called. Eventually, processing causes the BswM (task manager) to execute actions as well a trigger software. Any reconfiguration actions are under the control of the integrator.*

Nissan Rogue - AUTOSAR



AUTOSAR

Specification of ECU State Manager
V3.0.0
R4.0 Rev 3

7.1.1 STARTUP Phase

The purpose of the STARTUP phase is to initialize the basic software modules to the point where Generic Mode Management facilities are operational. For more details about the initialization see chapter *7.3*.

7.1.2 UP Phase

Essentially, the UP phase starts when the BSW Scheduler has started and BswM_Init has been called. At that point, memory management is not initialized, there are no communication stacks, no SW-C support (RTE) and the SW-Cs have not started. Processing starts in a certain mode (the next one configured after Startup) with corresponding runnables, i.e. the BSW MainFunctions, and continues as an arbitrary combination of mode changes which cause the BswM to execute actions as well as triggering and disabling corresponding runnables.

From the ECU Manager Module perspective, the ECU is "up", however. The BSW Mode Manager Module then starts mode arbitration and all further BSW initialization, starting the RTE and (implicitly) starting SW-Cs becomes code executed in the BswM's action lists or driven by mode-dependent scheduling, effectively under the control of the integrator.

Source: Specification of ECU State Manger, AUTOSAR Document No. 078, R 4.0 Rev 3